User Friendly Test Cases

Login Controller Tests

- **loginUser_ValidCredentials_Success:**
  - Tests logging in with valid credentials (email and password).
  - Mocks an existing user in the repository.
  - Expects a successful login response with a non-empty session token.
- **loginUser_InvalidCredentials_Error:**
  - Tests logging in with invalid credentials (email and password).
  - Mocks no user found in the repository, simulating an invalid login attempt.
  - Expects an error message: "Invalid email or password."
- **loginUser_AlreadyLoggedIn_Message:**
  - Tests logging in when the user is already logged in (has a session token).
  - Simulates a scenario where a session token is already present.
  - Expects a successful response but doesn't specify any unique content for already logged-in users (could be improved based on controller behavior)

Logout Controller Test

- **logoutUser_ValidSession_Success:**
  - Tests logging out with a valid session token.
  - Mocks a valid session token in the request.
  - Expects a successful logout response: "You have successfully logged out."
- **logoutUser_NoSessionToken_Success:**
  - Tests logging out without a session token (e.g., user manually accessing the logout URL).
  - Expects a successful response indicating the user has logged out, even without an active session token.

Signup Controller Test

- **signupUser_ValidUser_Success:**
  - Tests user signup with valid details (name, email, password, confirm password).
  - Mocks no existing user with the same email in the repository.
  - Expects a redirection on successful signup (common for redirecting to login or home page).
- **signupUser_EmailAlreadyExists_Error:**
  - Tests user signup with an email that already exists in the system.
  - Mocks an existing user with the same email.
  - Expects an error message on the signup page: "This email is already in use."

Medical Test Cases

Add Medical Reports Controller Test

- **addingMedicalReport_Success:**
    - Tests successfully adding a medical report when the user is logged in
    - Expect a successful addition of medical reports to the database

- **addingMedicalReport_MissingInput:**
    - Tests adding a medical report when the user is logged in and provides medical report with missing inputs
    - Expect an indication to fill in all required data before adding the medical report to the database

- **addingMedicalReport_Error_UniqueData:**
    - Tests adding a medical report when the user is logged and uses an existing petID and or pet name
    - Expect an error message indicating duplicate data for the same pet and reload add medical page

View Existing Medical Reports Controller Test

- **viewingMedicalReport_NoDataMessage:**
    - Tests viewing of medical reports when the user is logged in but has no data under the account
    - Expect a message saying there is no available data

- **viewingMedicalReport_Success:**
    - Tests viewing of medical reports when the user is logged in and has data
    - Expects the user to be able to select existing pets and view all data

Download Medical Report Controller Test

- **downloadMedicalReport_Success:**
    - Tests viewing of medical reports when the user is logged in and has data
    - Expects the user to be able to download the report in a PDF form

Educational Resources Test Cases:

EducationControllerTest.java

- **shouldFilterArticlesByTopic:** Verifies that when the topic parameter (e.g. exercise) is passed, the response contains only articles matching that topic.
- **shouldFilterArticlesByDateRange:** Verifies that when a date range (e.g. 2017-2020) is passed, the response contains articles published within that range.

GuidesControllerTest.java.

- **shouldFilterGuidesByTopic:** Verifies that when the topic parameter (e.g. Nutrition/Diet) is passed, the response contains only guides matching that topic.
- **shouldFilterGuidesByReadingTime:** Verifies that when the time parameter (e.g. 5-10 Minutes) is passed, the response contains only guides with the same estimated reading time range.

TrendsControllerTest.java

- **shouldContainTrendDates_Success:** Verifies that when the filter parameter is set to date, the response contains trends with specific dates (e.g. September 6, 2024).
- **shouldContainTrendRelevance_Success:** Verifies that when the filter parameter is set to relevance, the response contains trends related to specific relevant content (e.g. New Scholarship Program).

VideosControllerTest.java

- **shouldFilterVideosByTopic:** Verifies that when the topic parameter (e.g. Training) is passed, the response contains only videos matching that topic.
- **shouldFilterVideosByDuration:** Verifies that when the duration parameter (e.g. 0-5) is passed, the response contains only videos with a duration 0-5 minutes.

Appointment Scheduling Test Cases:

BookingController.java Tests:

1. **createBooking_ValidData_Success**
    - Description: Tests creating a booking with valid date and time slot.
    - Mock Behaviour: Mocks the time slot being available and calls to createBooking in BookingService.java.
    - Expected Outcome: The user is redirected to the “My Booking/s” page with the booking successfully created, which can be verified if the booking a user just selected appears on the “My Booking/s” page.
2. **cancelBooking_ValidBookingId_Success**
    - Description: Tests canceling a booking by its ID.
    - Mock Behaviour: Mocks a valid booking ID and calls to deleteBooking in BookingService.java
    - Expected Outcome: The booking is deleted successfully, and the user is redirected to the “My Booking/s” page with a success/confirmation message stating that their booking was successfully canceled.
3. **rescheduleBooking_ValidBookingId_Success**
    - Description: Tests rescheduling an existing booking by its ID.
    - Mock Behaviour: Mocks a valid booking ID and the deleteBookingById method.

- Expected Outcome: The booking is canceled/removed and the user is redirected to the “Make a Booking” page to make/reschedule to a new booking.

MyBookingsController.java Tests:

1. **getBookings_DisplayAllBookings_Success**
    - Description: Tests fetching all bookings and displaying them on the “My Booking/s” page.
    - Mock Behaviour: Mocks a list of Booking objects returned by BookingService.java
    - Expected Outcome: The bookings are added to the model, and the “My Booking/s” page is displayed with the bookings in the pre-specified list format.

ClinicController.java Tests:

1. **showClinicSelector_DisplayAllClinics_Success**
    - Description: Tests fetching all clinics and displaying them in the clinic selector drop-down/box of the clinicSelector.html page/file.
    - Mock Behaviour: Mocks a list of Clinic objects which are returned by the ClinicService.java file.
    - Expected Outcome: The clinics are added to the model, and the clinic selector page displays all the clinics including their name, price, and star-rating in the drop-down/box options.

Prescription Controller Test

- **getPrescriptionsAndHistory_ValidSessionToken_Success:**
    - Tests retrieving prescriptions and prescription history with a valid session token.
    - Mocks:
        - A valid session token associated with a user's email.
        - Empty prescription and prescription history lists returned by PrescriptionService and PrescriptionHistoryService.
    - Expects:
        - Status 200 OK.
        - The view name to be "prescriptions".
        - The isLoggedIn model attribute to be true.
        - The prescriptions and histories model attributes to contain empty lists (indicating no data for the user).
- **getPrescriptionsAndHistory_NoSessionToken_Failure:**
    - Tests retrieving prescriptions and prescription history without providing a session token.
    - Expects:
        - Status 200 OK.
        - The view name to be "prescriptions".
        - The isLoggedIn model attribute to be false (indicating the user is not logged in and no session token was provided).